www.icgst.com

Enhanced User Authentication through Keystroke Biometrics for Short-Text and Long-Text Inputs

M. AbdelDayem, H. Hemeda, A. Sarhan
Department of Computer and Control Engineering, Faculty of Engineering, Tanta University, Tanta, Egypt
mahmoudabdeldayem@hotmail.com, *hamed.hemeda@hotmail.com, amany_m_sarhan@yahoo.com*

Abstract
Password typing is the most common authentication method to secure computer systems. However, due to its simplicity, it could be easily compromised. Keystroke dynamics can be combined with password checking to result in a more secure and robust system, inexpensively and effectively. This paper investigates the use of the keystroke dynamics biometric as an authentication method. We employ algorithms to authenticate users through their short-text and long-text typing patterns. The experimental results show that our system has high specificity when dealing with short-text input and high sensitivity when dealing with long-text input.

Keywords: *Computer Security, Authentication, Biometrics, Keystroke Dynamics, Pattern Recognition, Identity Verification.*

Nomenclature

FAR	False Accept Rate
FRR	False Reject Rate
EER	Equal Error Rate
P-R	Press-Release
P-P	Press-Press
R-P	Release-Press
VKF	Virtual Key Force
SVM	Support Vector Machine
NN	Neural Networks
WNN	Weightless Neural Networks
GA	Genetic Algorithms
PSO	Particle Swamp Optimization
XML	Extensible Markup Language

1. Introduction
Computers are used to store and process sensitive and confidential information. It became necessary to secure this information from intruders. Authentication is the process of determining whether a user should be allowed access to a particular system or resource. Password typing is the most common authentication method for computer systems. However, it is vulnerable to imposter attacks. Therefore, other authentication methods, such as biometrics-based authentication methods, are used alone or integrated with password typing authentication.

Biometrics technologies provide promising means of identification and authentication. Biometrics is the science and technology of measuring and statistically analyzing biological data. Biometrics can be divided into two categories: physiological and behavioral [17]. Physiological characteristics refer to what the person is, such as fingerprint, face, iris, etc. Behavioral characteristics refer to what a person does, or how the person uses the body, such as voice, signature, keystroke dynamics, mouse dynamics, etc.

Keystroke dynamics is a process of analyzing the way a user types at a keyboard by monitoring it in order to identify the users based on habitual typing rhythm patterns. It is a very attractive method due to several reasons. Firstly, it is not intrusive and computer users frequently type on a computer keyboard. Secondly, it is inexpensive since the only hardware required is a computer and a keyboard. Thirdly, after an authentication phase has verified a user's identity, the user continues using the keyboard and that allows subsequent checking for the user's identity [5].

In this work, we propose a keystroke dynamics biometric system to verify the identity of users. We adopted different algorithms to authenticate users through their short-text and long-text data. We used a statistical method as our short-text algorithm and a k-nearest neighbor classification method as our long-text algorithm. The long-text algorithm depends on the relationship between users' data while the short-text algorithm depends on each user's data. We ran two experiments: the first one over a small set of eight users and the second one over a larger set of twenty six users.

The remainder of the paper is organized as follows: In the next section, we provide some general background information on keystroke dynamics authentication systems. Section (3) describes our proposed keystroke biometric system and the phases it constitutes. Section (4) describes our keystroke dynamics application, the experiments, presents the experimental results, and evaluates the performance of our system. Section (5) concludes the paper and provides suggestions for future work.

2. Related Work

Different features for classification were proposed by different studies. Latency between keystrokes is one of the most commonly used features by many researchers. There are three types of latencies as defined by Balagani et al. [4] - press-to-press (P-P), release-to-release (R-R) and release-to-press (R-P) latencies. The P-P latency is also defined as digraph and is used by most researchers. Trigraph is the time interval between the presses or releases of alternate keystrokes. Some researchers have also used other N-graph features for identification. The trigraph feature was first used by Bergadano et al. [6]. Key hold time or dwell time is defined as the time for which each keystroke was pressed and is used by many researchers as well. Robinson et al. [18] concluded that hold times are much more important than inter-key times. Certain types of keyboards are available in the market which can measure the pressure applied to a key while typing. Attempt to recognize emotion from users typing pattern using pressure sensitive keyboards was carried out by Lv et al. [15] A new feature called Virtual Key Force (VKF) was proposed by Shanmugapriya and Padmavathi [19]. It is calculated based on the typing speed and behavior of the user on the keyboard.

Different classification methods were used by different studies. Some studies used simple algorithms while others used more complex algorithms. Table 1 lists the different classification methods undertaken by researchers towards developing authentication and identification systems. The studies evaluate their systems' performances using False Accept Rate (FAR) and False Reject Rate (FRR) or Equal Error Rate (EER). FAR and FRR are described in Section 4.4. EER is the rate at which both accept and reject errors are equal.

We found that most previous keystroke biometric studies focused on short-text input, while a few studies focused on long-text input. No study has incorporated both short-text and long-text authentications before. The relationship between the two performance evaluation metrics (FAR and FRR; both are described in Section 4.4) is inversely proportional; when one decreases, the other increases. Thus, both metrics cannot be improved for an authentication system at the same time, as shown in table 1, without using sophisticated, computationally intensive algorithms. In this paper, we combine both short-text and long-text authentications and work on improving one performance metric for each authentication type while using simple algorithms. Another advantage that we gain from combining both authentications is that long-text authentication is very appropriate for dynamic authentication, where the user identity could be verified through his interaction with the secured system after he is authenticated, while short-text authentication is used for static authentication, where the user identity is verified during login. Moreover, we developed our own java application (see Section 4.1) and collected the data from our users instead of using readily available keystroke databases on the Internet [20, 27]. This gave us more flexibility in extracting the features and the information we need from our users to achieve the best performance for our system.

Study	Classification Technique or Method	Results		
		FAR	FRR	EER
[10]	Distance measure	8.33%	3.33%	
[14]	Statistical	4.5%	5.5%	
[1]	Weight measure	22.5%	0%	
[2]	Neural Networks (NN)	0%	5.01%	
[22]	Weighted NN			
[20]	Parallel decision trees	0.88%	9.62%	
[7]	Gaussian Mixture Model			1.3% 5.88%
[9]	Support Vector Machines (SVM)			13.45%
[16]	Statistical			12.7%
[3]	SVM with Genetic Algorithms (GA) and Particle Swarm Optimization (PSO)	0.43%	4.75%	
[13]	Mean Nearest Neighbor-Multi Layer Perceptron (NN-MLP)			9.96% 11% 16.24%
[5]	Random Forest	1%	14%	

Table 1. Authentication and identification using different classification techniques.

3. Proposed Keystroke Biometric System

The proposed keystroke biometric system consists of the following phases: keystroke data collection and storage, feature extraction, text processing, and pattern classification and login phase. Figure 1 shows a flow diagram of the phases that form the proposed keystroke biometric system. The next subsections will describe each phase.

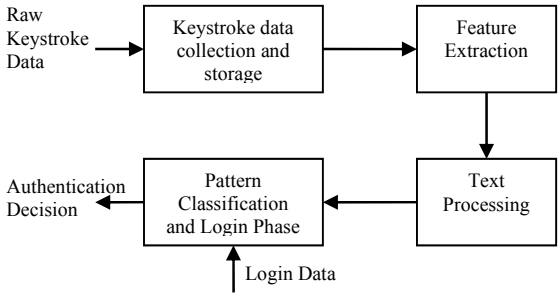

Figure 1. Flow diagram for the phases of the proposed keystroke biometric system.

latencies (described in Section 3.2.1) for both short-text and long-text inputs, but processed them differently in the text processing phase, to extract short-text and long-

text features. We modified the long-text hierarchy tree for keys durations (described in Section 3.2.2) by modifying the non-letters feature branch. We added more features for keys ratios for long-text authentication (described in Section 3.2.2). To improve the performance of our system, we added a parameters optimization phase for both short-text and long-text processing (see Sections 3.3.1.2 and 3.3.2.3). The significance of keys durations and transitions features for short-text authentication and keys durations, transitions and ratios features for long-text authentication were also tested to decide which features are the most appropriate for each authentication (see Sections 3.3.1.2, 3.3.2.3, and 4.3).

3.1. Keystroke data collection and storage phase

The keys timing information is collected for each user for each key using key listeners in Java. This information is used to extract the features in the next phase. For every key pressed, the following information is collected: Key's character or name, Key's location on the keyboard (whether it is left, center, or right), Times when the key was pressed and released (in milliseconds).

3.2. Feature Extraction Phase

In this phase, the required features are extracted from the raw keystroke timing information for both short-text and long-text inputs. The ways in which the short-text and long-text features are extracted are different. They are detailed in the next subsections.

3.2.1. Short-text Features

The following latencies [18] are calculated from the raw keystroke timing information and stored for further processing:

- Key Duration: It is the time interval between pressing a key and its release. It is sometimes called **Press-Release (P-R) latency**, **dwell**, or **hold time**.
- Key Transition, T_1: It is the time interval between pressing two successive keys. It is sometimes called **Press-Press (P-P)** or **digraph latency**.
- Key Transition, T_2: It is the time interval between releasing a key and pressing the next. It is sometimes called **Release-Press (R-P) latency** or **flight time**.

3.2.2. Long-text Features

The same latencies used for the short-text input are essentially used for the long-text input. They are computed for each character in the long-text data then the means and standard deviations for these latencies are calculated. They are then arranged in hierarchies for durations and transitions [21] as shown in Figures 2 and 3 respectively.

These hierarchy trees make use of the letter and digraph frequencies in English text. The left letters feature comprises the letters that are struck with the left hand (q, w, e, r, t, a, s, d, f, g, z, x, c, v, b) and the right letters feature comprises the letters that are struck with the right hand (y, u, i, o, p, h, j, k, l, n, m). The non-letters feature comprises two nodes: punctuation and numbers for the most frequent punctuation used in the English language and numbers, and modifiers for the frequent modifiers used in the English language such as shift, space, backspace, enter, etc.

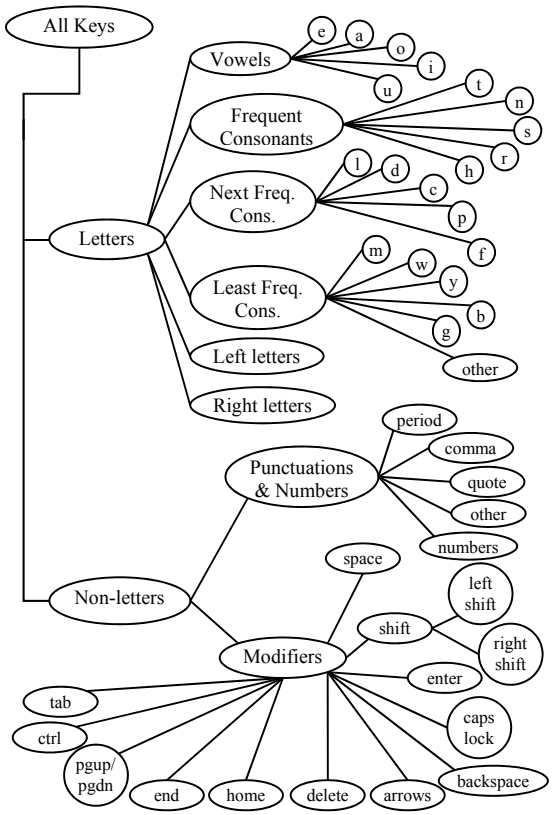

Figure 2. Hierarchy tree for all keys durations. Each category is represented by a mean and a standard deviation.

A feature vector, containing 251 features, is created for all means and standard deviations for durations (features 1-100), T1 (features 101-166), and T2 transitions (features 167-232). It also contains some useful ratios for key presses (features 233-246). These ratio features are useful to capture the users' preferences for using certain keys or key groups when they are typing, editing their text, and correcting their errors. The feature vector contains ratios of left mouse clicks, right mouse clicks, and double clicks (features 249-251). Finally, two other features are added as well (features 247-248). The first is the unadjusted keyboard input rate which equals the total time required to enter the text over the total number of key and mouse presses, and the second is the adjusted keyboard input rate which equals the total time required to enter the text minus any P-P time that is greater than 500 milliseconds (this represents the times the user is interrupted) over the total number of key and mouse presses.

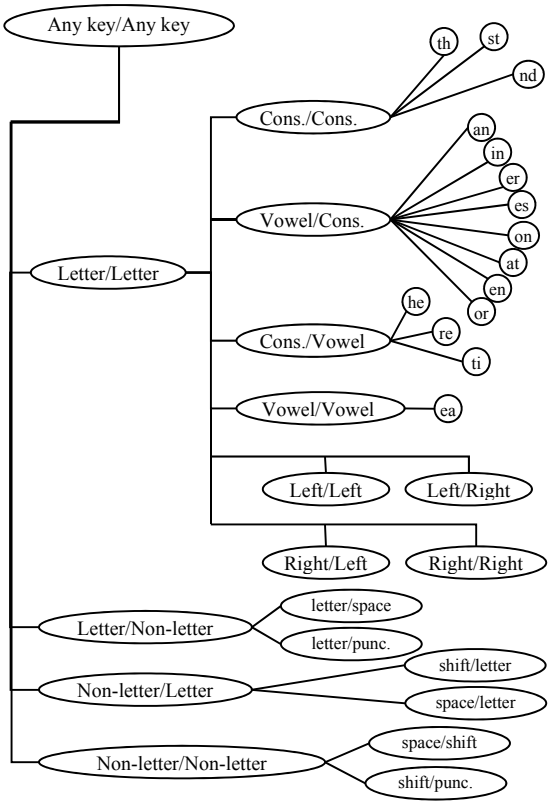

Figure 3. Hierarchy tree for all keys transitions [21]. Each category is represented by a mean and standard deviation for T_1 and T_2 transitions.

3.3. *Text Processing Phase*

Each user should enter his data many times to use them for training. Both short-text and long-text data are processed differently to prepare them for classification.

3.3.1. *Short-text Processing*

Short-text processing is done in two steps: calculating means, standard deviations, and weights for letters' durations and transitions, and optimizing short-text parameters for classification.

3.3.1.1. *Mean, Standard Deviation, and Weight calculation*

All samples for each user are compared with each other. Any redundant letter is removed. This will make all samples have the same length and letters.

The **mean**, μ_i, and **standard deviation**, σ_i, for each letter duration and transition are computed. **Weights** are then computed for each duration and transition over all samples as follows:

$$weight_i = \left(\left(\frac{\mu_i}{\sigma_i}\right) \times \frac{1}{\sum_{j=1}^{n} \left(\frac{\mu_j}{\sigma_j}\right)} \right) \qquad (1)$$

It calculates the ratio of μ_i/σ_i for each duration or transition i to the summation of all μ/σ for all durations or transitions, where n is the number of samples. A short-text profile is created for each user containing mean, standard deviation, and weight for each letter duration and transition for both username and email. This profile is compared to login data supplied by the user during authentication and the system decides whether to accept login or not accordingly.

3.3.1.2. *Parameters Optimization*

As the short-text authentication system is built, we create some constants. Instead of assuming specific values for these constants, it will produce much better results if we assume them as variables and optimize them in their reasonable ranges. We can get the best accuracy, sensitivity and specificity by optimizing these parameters. Moreover, using variables for the parameters will make our system more generic and can deal with any set of users as the parameters will be optimized according to the data the users supply.

The short-text parameters are optimized using a *hill climbing* method. Hill climbing [21] is a simple local search optimization technique. Since the optimized parameters have linear nature, hill climbing is very convenient to find their optimal values. Hill climbing works as follows: we assume a certain value for one of the parameters and run and test the system then, we change the parameter value and run and test the system while fixing the values of all the other parameters in the system. The value that produces the best performance for the system is considered the optimal value for the optimized parameter.

There are two types of testing to test the authentication system: legitimate data testing and imposter data testing. In legitimate data testing, the system is trained using the enrolled data and tested using a leave-one-out procedure on the enrolled data. The optimal parameter value should result in highest acceptance percentages in this type of testing implying few false rejects (Lowest FRR). In imposter data testing, the system is trained using the enrolled data and tested on imposter data supplied by the users logging in using each other's username/email. The optimal parameter value should result in lowest acceptance percentages in this type of testing implying few false accepts (Lowest FAR).

The short-text parameters that are optimized are k_d, k_t, τ_1, τ_2, $\%added_d$, $\%added_t$ and *acceptance%*. k_d and k_t are optimized in the range 1.0 to 4.0, in increments of 0.5, τ_1 is optimized in the range 0.3 to 0.8, in increments of 0.05 and τ_2 is optimized in the range 0.5 to 0.85, in increments of 0.05. $\%added_d$ and $\%added_t$ are optimized together in the ranges 0 to 1.0 and 1.0 to 0 in increments and decrements of 0.05 respectively so that their sum is always 1, and *acceptance%* is optimized in the range 0.2 to 0.8, in increments of 0.05. We have chosen these values for the optimization ranges because outside these ranges the authentication system will fail to work properly and will result in high percentages of false accepts or false rejects. For example, if we assumed the *acceptance%* equals 0.9, it means we need to have a 90% matching between login data and enrolled data to authenticate users which will result in many false rejects.

On the other hand, if we assumed the *acceptance%* equals 0.1, it means we have to have a 10% matching between login data and enrolled data to authenticate users which will result in high number of false accepts. This optimization procedure is repeated until the system produces the best results for both legitimate and imposter data testing.

The parameters cannot be optimized unless data by all users is supplied. Initial values of 1.0, 1.0, 0.65, 0.5, 0.5, 0.5 and 0.5 respectively are assigned to the parameters. We chose these initial values because these are median values that allow the system to perform well regardless of the set of users using it. We will describe the short-text parameters and their functions in Section 3.4.1.

3.3.2. *Long-text Processing*

Many processing steps are carried out on long-text data to prepare it for classification. They are described in the next subsections.

3.3.2.1. *Revised Mean and Standard Deviation Calculation*

If a key duration or transition exists in the long text very few times, it requires special handling when computing its mean and standard deviation. We make use of the parent nodes in the hierarchy trees for keys durations and transitions (see Figures 2 and 3) and use a fallback procedure to compute revised mean and standard deviation. This procedure is similar to the *"backoff"* procedures used in natural language processing [12].

So when the number of instances for a letter duration or transition is less than $\tau_{fallback}$, which is an optimized parameter, revised mean and standard deviation should be calculated [12]:

$$\mu' = \frac{n \times \mu + w_{fallback} \times \mu(fallback)}{n + w_{fallback}} \quad (2)$$

$$\sigma' = \frac{n \times \sigma + w_{fallback} \times \sigma(fallback)}{n + w_{fallback}} \quad (3)$$

Where μ' and σ' are revised mean and standard deviation, μ and σ are unrevised mean and standard deviation, n is the number of instances, $\mu(fallback)$ and $\sigma(fallback)$ are fallback mean and standard deviation, and $w_{fallback}$ is another optimized parameter that converges the unrevised mean and standard deviation to the fallback mean and standard deviation in the fallback procedure. The fallback means and standard deviations are determined by the parent nodes in the hierarchy trees in Figures 2 and 3. For example, if the letter 'g' exists a few times (less than $\tau_{fallback}$) in the long text, its revised mean and standard deviation are calculated using the fallback mean and standard deviation of least frequent consonants, provided that they exist enough times in the long text.

3.3.2.2. *Outlier Removal*

Outlier removal is very significant because a keyboard user could get interrupted while typing or pause for whatever reason, and the resulting outliers (usually overly long transition times) could skew the feature measurements. If any duration or transition is more than k_σ standard deviations from their respective mean, it is considered an outlier and is removed. k_σ is another optimized parameter. After all outliers are removed, means and standard deviations should be recalculated. Revised means and standard deviations should be recalculated, if needed. Outlier removal is performed *iterations* times, where *iterations* is an optimized parameter.

3.3.2.3. *Parameters Optimization*

We assume the parameters created in our long-text authentication system as variables instead of constants and optimize them in their reasonable ranges for the same reasons stated in Section 3.3.1.2 for short-text data parameters. The long-text data parameters are optimized using a *hill climbing* method in the same manner as the short-text data parameters. The optimization process works in the same manner described in Section 3.3.1.2.

There are two types of testing to test the authentication system: legitimate data testing and imposter data testing. The system is trained using the enrolled data and tested using a leave-one-out procedure on the enrolled data for both types of testing. For legitimate data testing, the optimal parameter value should result in highest acceptance percentages implying few false rejects (Lowest FRR). For imposter data testing, the optimal parameter value should result in lowest acceptance percentages implying few false accepts (Lowest FAR).

The long-text parameters that are optimized are $\tau_{fallback}$, $w_{fallback}$, k_σ, *iterations*, *#neighbors*, w_d, w_t and w_r. $\tau_{fallback}$ is optimized in the range 3 to 10, in increments of 1, $w_{fallback}$ is optimized in the range 1 to 5, in increments of 1, k_σ is optimized in the range 1.0 to 4.0, in increments of 0.5, *iterations* is optimized in the range 0 to 4, in increments of 1, and *#neighbors* is optimized in the range 2 to 12, in increments of 2. w_d, w_t and w_r are optimized together so that their sum must equals 1.0. w_t is optimized in the range 1.0 to 0, in decrements of 0.05, and w_r is optimized in the range 0 to 0.3, in increments of 0.05, while w_d will become $1.0 - (w_t + w_r)$.

The parameters cannot be optimized unless data by all users is supplied. Initial values of 5, 2, 1.0, 1, 10, 0.3, 0.55 and 0.15 respectively are assigned to the parameters. $\tau_{fallback}$, $w_{fallback}$, k_σ and *iterations* are described in the previous sections (Secs. 3.3.2.1 and 3.3.2.2). We will


describe the remaining long-text parameters and their functions in Section 3.4.2.2.

3.3.2.4. *Features Normalization*

The features used have different units and ranges: the means could be positive or negative (in case of overlapping T_2 transition); the standard deviations are always positive and their absolute values are usually much less than the means; and the ratios are small fractions. Since we are using distance-based classification method, the significant differences stated above will have big influence on the distance. This will have a very undesirable effect on our system. To eliminate this and to give each measurement roughly equal weight, all the feature values should be normalized to the range 0.0 to 1.0. Means, standard deviations, and ratios are normalized using the following formula:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (4)$$

Where x' is the normalized value, x is the value to normalize, and x_{min} and x_{max} are its smallest and largest values respectively over all samples from all users [8].

3.4. Pattern Classification and Login Phase

In this phase, different classification techniques are used for short-text and long-text data, and login data is compared to decide whether the user is authenticated or not.

3.4.1. Short-text Login Phase

When a user tries to log in, he enters his username and email. Both are compared with his short-text profile. We take the letters' durations for the username as an example. Every letter's duration is compared with its corresponding one in the profile to determine whether it falls in the $\mu \pm k_d \sigma$ range, where k_d is an optimized parameter that controls the strictness of the $\mu \pm \sigma$ range; the bigger the value of the parameter, the more loose the range will be. If the letter's duration falls in the $\mu \pm k_d \sigma$ range, the letter duration's weight is added to the total weight of the login try for durations and the number of matches for the login try is incremented by one. When the comparison is performed for all letters' durations of the user's username, the total weight is compared with τ_1 and the number of matches divided by the total number of characters in the username is compared with τ_2. τ_1 is an optimized parameter that represents the minimum weight value to consider a matching between login and enrolled data while τ_2 is an optimized parameter that represents the minimum accepted value for (matches/characters). If the total weight is greater than τ_1 or the number of matches counted over the total characters is greater than τ_2, the value $\%added_d$ is added to the login try acceptance value for the username.

The same comparison is then done for all letters' transitions for the user's username using the parameter k_t and adding the value $\%added_t$ to the login try acceptance value for the username, where k_t is an optimized parameter that controls the strictness of the $\mu \pm \sigma$ range for letters' transitions. $\%added_d$ and $\%added_t$ are optimized parameters that determine which of the durations and transitions features are more effective in short-text authentication. For instance, if $\%added_d$ equals 0.9 and $\%added_t$ equals 0.1, this means that letters' durations matching between login and enrolled data adds 90% to the login try acceptance value while letters' transitions matching between login and enrolled data adds only 10% to the login try acceptance value.

The same comparison is then held for all letters' durations and transitions for the user's email yielding a login trial acceptance value for the email. The login trial acceptance value is the average of both username and email values. The login trial is considered successful if the login trial acceptance value is greater than or equal to *acceptance%*, which is an optimized constant that represents the minimum acceptance percentage to deem the login trial acceptable.

3.4.2. Long-text Pattern Classification and Login Phase

A Nearest Neighbor classifier [11] is used for verification in our system. Nearest neighbor classifier is a distance-based classifier that uses *Euclidean distance* between the feature vectors of test and training data to verify the identity of the user.

3.4.2.1. Dichotomy Transformation Process

Our system is not concerned with user identification. It is only concerned with user verification. The authentication problem is a multi-class (*polychotomy*) problem therefore; it will get much easier if we transformed it to a two-class (*dichotomy*) problem [21]. The transformation is done by calculating vector distances between pairs of samples of the *same* user (*intra-user distances*, denoted by x_{intra}) and pairs of samples of *different* users (*inter-user distances*, denoted by x_{inter}) as follows:

$$x_{intra} = (d_{ij}, d_{ik}) = |d_{ij} - d_{ik}|, i = 1 \; to \; n, \; and \; j, k = 1 \; to \; m, j \neq k \quad (5)$$

$$x_{inter} = \delta(d_{ij}, d_{kl}) = |d_{ij} - d_{kl}|, i, k = 1 \; to \; n, \; i \neq k, and \; j, l = 1 \; to \; m \quad (6)$$

Where n is the number of users, m is the number of samples, and d_{xy} is the vector of values for all features of sample y supplied by user x. The *intra-user* and *inter-user distances* are calculated as the absolute differences between means and standard deviations for durations, transitions, and ratios of all letters in enrolled data.

3.4.2.2. Long-text Login Phase

When the user logs in, the *login intra-user distances* are calculated between the login data and corresponding enrolled data. Since we are using a distance-based classification method, Euclidean distances are calculated for durations, transitions and ratios between: *Login intra-user distances* and *Enrolled intra-user distances*, and *Login intra-user distances* and *Enrolled inter-user distances*.

Euclidean distance is the square root of the sum of squared distances. We use a k-nearest neighbor classifier. *#neighbors* is an optimized parameter that represents the number of neighbors required for classification. The Euclidean distances for durations, transitions and ratios are compared together and according to *#neighbors* the matching percentages are calculated for durations, transitions and ratios. To calculate the login trial acceptance percentage for the long-text, the matching percentages are multiplied by the constants w_d, w_t and w_r respectively as follows:

$$acceptance\% = w_d * matching\%_d + w_t * matching\%_t + w_r * matching\%_r \quad (7)$$

Where $matching\%_d$, $matching\%_t$ and $matching\%_r$ are the matching percentages calculated for durations, transitions and ratios, *acceptance%* is the login trial acceptance percentage, and w_d, w_t and w_r are optimized parameters that represent weights for durations, transitions and ratios matching percentages. These weights determine which of the durations, transitions and ratios features are more effective in long-text authentication. For instance, if w_d equals 0.7, w_t equals 0.3 and w_r equals 0.0 (remember that the sum of the three parameters must equals 1.0) this means that *acceptance%* now equals ($0.7 \times matching\%_d + 0.3 \times matching\%_t$) giving a 70% weight for durations features, 30% weight for transitions features and no weight to ratios features. The login trial is considered successful if the login trial acceptance percentage is greater than or equal to 50%.

4. Experiments

To verify the effectiveness of the proposed system, two sets of experiments were carried out; the first time was carried out for eight users and the second time was carried out for twenty six users (including the first eight users). Each user supplied five samples for training and two samples for testing. Some users supplied samples for imposter testing.

4.1. Keystroke Dynamics Application

We developed a Java application (see Figure 4) to collect the keystroke data. The users are required to type in their name and email in their respective fields. We first required the users to type to type username/password combinations but found it difficult to make users type their real passwords in our application for confidentiality reasons, and it will deteriorate the reliability of our system if they typed dummy passwords they are not used to. The username and email that users type should be at least 20 characters combined. The username/email combination forms the short-text input. The user is required to copy a predefined long-text [21] for training (see Text 1 in Appendix) into a text field. It contains 127 words (632 characters). For testing, the users either copied the same long-text used in training or another text [21] that has rather similar components (see Text 2 in Appendix). This text contains 107 words (556 characters).

To help the users keep track of the number of times they entered their data to the application, we used a submission number, as shown in Figure 4, which is automatically incremented according to the number of samples they have submitted before. The users should enter short-text data without errors or pause.

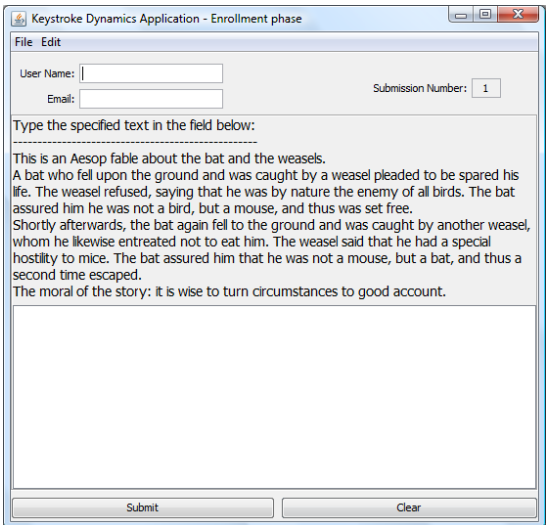

Figure 4. Keystroke Dynamics Application

Users are not allowed to make errors when enrolling their short-text data. Errors are allowed during the login phase only but not during data enrolment. Users are allowed to type in their names and emails sequentially, therefore backspaces are allowed, but not inserting letters in the middle of the words. The user is free to type long-text data in his normal typing pattern. He can delete, edit, and do whatever suits him. Whenever a user is not comfortable with his typing, or had an error during enrolling his short-text data, he can clear the data, by

pressing the clear button, and starts typing again. When the user finishes typing, he presses the submit button. When the data is submitted, it is stored in three XML files. The first file contains the raw keystroke timing information for username, email, and long-text data, the second file contains the short-text latencies for the username and email, and the third file contains the features for long-text data.

4.2. *Data Collection and Environmental Conditions*

We collected the keystroke data from the volunteers instead of using available keystroke data on the internet. Users were asked to enter their data five times. They were asked to leave at least 12 hours between entering training samples, so that a user does not adopt a certain typing pattern that he can't apply afterwards when logging in. Also, the samples need to be spread out over time similar to what might be expected in an actual application environment.

All users used laptops for entering the data. They used the same keyboard during training and testing. Twenty three users used their personal computer and keyboard for entering the data while three users didn't use their personal computers and keyboards. Also, the users were asked to be relaxed when typing their samples and so they were given the java application to allow them to type at home when they are in good mood.

Table 2 shows some information about the users who participated in the experiments. We represent the users with their initials (first column in the table). The first eight users participated in both experiments while the remaining users participated in the second experiment only. The typing speed is estimated for the average time the users took to enter their long-text data. The error rate is estimated for the average number of characters typed by the users to correct their errors (backspace, arrows and delete keys). Typing speed and error rate values are displayed in the table in brackets in their respective columns. Note that the levels for typing speed and error rate in the table are estimated for the users with respect to each other.

4.3. Experimentation and Optimization Results

In this section, we describe the two experiments and present the optimization results for each.

4.3.1. First *Experiment (8 users)*

All the eight users entered Texts 1 and 2 for testing (see Appendix). Six users were asked to try to log in using the other users' username/email and Text 1 (the same long-text used by all users in training) to provide imposter data (48 samples). They were asked to type normally. They didn't try to employ other users' typing pattern.

The optimization process for short-text and long-text data parameters for eight users resulted in the values shown in Table 3. For short-text data parameters, the values for $\%added_d$ and $\%added_t$ of 0.0 and 1.0 respectively mean that we relied **only** on keys transitions features for short-text authentication for eight users and keys durations features were redundant. The value for *acceptance%* of 0.55 means that the system needs 55% matching between login data and enrolled data to consider the authentication successful for this set of users.

User	Gender	Age	PC Brand and Model	PC?	Typing Speed	Error Rate
AM	Male	29	Macbook Pro 2012	Yes	Fast (2'51")	Very High (78)
AS	Male	21	Toshiba Satellite A350-13A	No	Medium (4'27")	High (36)
MA	Male	28	HP Pavillion G Series G6 584032-001	No	Medium (5'55")	Medium (22)
KD	Male	29	Toshiba Satellite A100-813	Yes	Medium (4'18")	Very Low (6)
KK	Male	28	HP Pavillion G Series G6-1162ex	Yes	Medium (5'40")	Very Low (8)
MD	Male	27	Toshiba Satellite A350-13A	Yes	Fast (3'19")	Very High (73)
MS	Male	19	Toshiba Satellite A350-13A	No	Slow (6'58")	High (43)
MG	Male	27	HP Pavillion G Series G6-1236ee	Yes	Medium (5'36")	Medium (25)
AB	Male	25	Dell Inspiron 15R (N5110)	Yes	Slow (7'11")	Medium (27)
AY	Female	48	Acer	Yes	Medium (4'57")	Medium (27)
AA	Male	21	Dell Inspiron 15	Yes	Medium (5'30")	Low (11)
BG	Female	28	Dell Inspiron 15	Yes	Slow (6'39")	Medium (26)
DM	Female	27	HP	Yes	Slow (7'43")	Medium (21)
FA	Female	25	HP Pavilion G series G6	Yes	Medium (5'56")	Low (17)
GK	Male	25	Lenovo z510	Yes	Fast (2'44")	High (36)
GS	Female	21	HP Pavilion G Series G6	Yes	Very Slow (14'54")	Medium (27)
KH	Female	21	HP Pavilion G Series G6	Yes	Slow (8'53")	Very High (91)
ML	Female	26	Dell Inspiron N5010	Yes	Medium (4'59")	Low (12)
KO	Male	21	HP	Yes	Fast (3'22")	Medium (22)
MB	Female	28	Dell Inspiron 15-3537	Yes	Slow (6'57")	Low (12)
ME	Female	23	Fujitsu AH530 Lifebook A Series	Yes	Very Slow (12'25")	Low (11)
SL	Male	21	Fujitsu Lifebook S760	Yes	Fast (3'30")	High (39)
RS	Female	25	Lenovo SL500 Thinkpad	Yes	Medium (4'10")	Very Low (6)
SD	Male	21	Dell Inspiron 15 3521	Yes	Slow (7'2")	Medium (24)
TA	Male	23	MSI CR620	Yes	Very Fast (1'58")	Low (12)
TY	Female	21	HP EliteBook 8440p	Yes	Slow (6'36")	High (45)

Table 2. Information about the users who participated in the experiments.

For long-text data parameters, the value for *iterations* of 0 means that the outlier removal process in the long-text processing phase was not required for this set of users. The values for w_d, w_t and w_r of 1.0, 0.0 and 0.0 respectively mean that we relied **only** on keys durations features for long-text authentication for eight users and keys transitions and ratios features were redundant.


Parameter Name	Parameter Value	Parameter Name	Parameter Value
k_d	1.0	$\tau_{fallback}$	**3**
k_t	1.5	$w_{fallback}$	**1**
τ_1	0.45	k_σ	**1.0**
τ_2	0.65	***iterations***	**0**
$\%added_d$	0.0	***#neighbors***	**6**
$\%added_t$	1.0	w_d	**1.0**
acceptance%	0.55	w_t	**0.0**
		w_r	**0.0**

Table 3. Short-text and long-text data parameters optimization results for eight users.

4.3.2. Second *Experiment (26 users)*

All the twenty six users entered Texts 1 and 2 for testing (see Appendix). The same six users were asked to try to log in using the other users' username/email and Text 1 (the same long-text used by all users in training) to provide imposter data (156 samples).

The optimization process for short-text and long-text data parameters for twenty six users resulted in the values shown in Table 4. For short-text data parameters, the values for $\%added_d$ and $\%added_t$ of 0.05 and 0.95 respectively mean that we relied by 95% on keys transitions features and by only 5% on keys durations features for short-text authentication for twenty six users. The value for *acceptance%* of 0.5 means that the system needs 50% matching between login data and enrolled data to consider the authentication successful.

For long-text data parameters, the value for *iterations* of 1 means that the outlier removal process in the long-text processing phase is required to be performed one time for this set of users. The values for w_d, w_t and w_r of 1.0, 0.0 and 0.0 respectively mean that we relied **only** on keys durations features for long-text authentication for twenty six users and keys transitions and ratios features were redundant, as the case with the first set of users.

Short-text Parameters		Long-text Parameters	
Parameter Name	Parameter Value	Parameter Name	Parameter Value
k_d	1.5	$\tau_{fallback}$	4
k_t	1.0	$w_{fallback}$	5
τ_1	0.7	k_σ	2.0
τ_2	0.55	***iterations***	1
$\%added_d$	0.05	***#neighbors***	4
$\%added_t$	0.95	w_d	1.0
acceptance%	0.5	w_t	0.0
		w_r	0.0

Table 4. Short-text and long-text data parameters optimization results for twenty six users.

4.4. Performance Evaluation

The most commonly used metrics to evaluate the performance of a keystroke biometric system or any other biometric system are False Rejection Rate (FRR) and False Acceptance Rate (FAR). FRR is the percentage of legitimate users who are labeled as imposters and denied access. FRR denotes the system's specificity. FAR is the percentage of imposters who are allowed access to the system. FAR denotes the system's sensitivity. Accuracy is the percentage of legitimate users who are allowed access. It equals (100 – FRR). The average matching percentages for legitimate and imposter data are the total matching percentages for all users over the total attempts for legitimate and imposter data respectively. Tables 5 and 6 show the performance of our keystroke biometric system for both short-text and long-text data for eight users and twenty six users respectively.

	Short-text data	Long-text data		
		Using Text 1	Using Text 2	Total
Average matching % for legitimate data	100%	100%	89.6%	94.8%
Average matching % for imposter data	25%	2.4%		
Accuracy	**100%**	100%	87.5%	**93.8%**
FRR	**0%**	0%	12.5%	**6.2%**
FAR	**12.5%**	**2.1%**		

Table 5. Performance of our keystroke biometric system for both short-text and long-text data for eight users.

	Short-text data	Long-text data		
		Using Text 1	Using Text 2	Total
Average matching % for legitimate data	84.4%	87.5%	45.2%	66.4%
Average matching % for imposter data	22.5%	1.12%		
Accuracy	**100%**	96.2%	61.6%	**78.9%**
FRR	**0%**	3.8%	38.5%	**21.1%**
FAR	**27.6%**	**1.3%**		

Table 6. Performance of our keystroke biometric system for both short-text and long-text data for twenty six users.

As shown in Table 5, for short-text authentication for the first set of users, all legitimate users were authenticated correctly and 12.5% imposter tries were considered successful. For long-text authentication, 2.1% imposter tries were considered successful. All legitimate users were authenticated correctly when using the same long-text data for training and testing, and 12.5% legitimate tries were denied access when using different long-text data for training and testing.

As shown in Table 6, for short-text authentication for the second set of users, all legitimate users were also authenticated correctly but a higher percentage of imposter tries of 27.6% were considered successful. For long-text authentication, only 1.3% imposter tries were considered successful and 21.1% of legitimate tries were denied access. When using the same long-text data for training and testing, only 3.8% of legitimate tries were denied access.

4.5. Results and Discussion

It is difficult to give a meaningful comparison of our approach with that of other studies as there is no unified data set under which the approaches can be compared and also no previous study has incorporated both short-text and long-text authentications before. We tested our


system's performance for short-text and long-text authentications over two sets of users, eight users and twenty six users, to emulate how our system would perform for a system of small number and larger number of users.

Firstly, for short-text authentication, our system had an FRR of 0% for both sets of users which means that no user who knows his login data (username/password combination) will be denied access to the system. Moreover, FAR of 12.5% and 27.5% for the two sets of users mean that some imposter tries could be accepted. Since keystroke dynamics is not the only mechanism for authentication but is integrated with password typing, so now an imposter needs to know the user's password and try to simulate his typing pattern too to gain access into the system.

For long-text authentication, FAR of 2.08% and 1.28% for the two sets of users mean that very few imposter tries would be allowed to gain access to the system. Using the same long-text for training and testing has very low FRR (0% and 3.84% for the two sets of users) which enhances the system's performance. However, making the users type the same long-text each time can cause inconvenience and that's why we needed to test the system performance when using different long-text data for training and testing. The system then had higher FRR values especially for the second set of users. The reason for the high rejection rate for the large set of users is the inconsistent and erratic typing patterns of some of our users when they have to copy a long-text they are not familiar with. This could be because English is a second language for them and some of them are inexperienced in English typing. This is clearly shown in the typing speed column of Table 2 where 15 users took more than 5 minutes to type a 600 character paragraph.

Our system showed that short-text authentication and long-text authentication are complementary to build a strong authentication system. Short-text authentication has higher FAR while long-text authentication has higher FRR, so each authentication type compensate for the shortcomings of the other. For instance, let's picture this scenario: When a user logs in, he provides his login information. Since our short-text authentication system has low FRR and high FAR, the user will mostly be allowed access to the system even when sometimes it's an imposter impersonating the user. Then, when the user is accessing the secured system, his long-text typing will be monitored and checked. Since our long-text authentication system for different long-text data in testing and training has low FAR and high FRR, the imposters will be detected (even when sometimes it's the legitimate user) and asked to enter the predefined long-text data (the same long-text data users entered during training) before they can continue accessing the secured system. Since the long-text authentication system has low FRR and FAR, imposters will be denied access to the system while legitimate users will be authenticated and their access to the secured system will be resumed.

Moreover, combining short-text and long-text authentications overcomes the inverse proportional relationship between FAR and FRR that most previous studies, as stated in Section 2, failed to overcome when they implemented single authentication systems that optimize one of the performance metric at the expense of the other, while using simple algorithms that are easy to implement and not computationally intensive.

The parameters optimization phase proved very important to our system. By tweaking the parameter values, we achieved the best performance for our system. The parameter optimization process also helped deciding which features are useful for short-text and long-text authentications. The optimization process for the short-text authentication system showed that we can rely solely on keys transitions features for the eight users system and by 95% for the twenty six users system. The optimization process for the long-text authentication system showed that we can rely solely on keys durations features for both eight users and twenty six users system and keys transitions and ratios features were redundant.

Furthermore, our two experiments showed that increasing the number of users reduced the system performance. For short-text authentication, FAR increased by a large value while FRR didn't change. For long text authentication, FRR increased by a very small value when using the same long-text data for training and testing and by a large value when using different long-text data for training and testing while FAR didn't change more or less.

All of the users, except for one, are in their twenties with an average age of about 25 years. 15 users are males and 11 users are females. For their long-text authentication testing, 60% of the male users got all their legitimate testing attempts accepted while 40% of them got only one attempt accepted. 63.6% of the female users got all their legitimate testing attempts accepted while 27.3% of them got only one attempt accepted and 9.1% got all attempts rejected. It appears that gender doesn't influence the accuracy of our system and it is more dependent on the users' typing experience, which can be identified by the time the users take to type the long-text data; the less time a user takes, the more experienced they are regardless of their typing errors, age or gender. Typing experience indicates less erratic typing style and hence more consistent typing patterns that allow better performance for our system.

5. Conclusions and Future Work

Keystroke dynamics is an inexpensive, non-intrusive biometric system that shows promising means for authentication for security systems. In this paper, we implemented a keystroke dynamics authentication system that deals with both short-text and long-text inputs to verify users identities. We used a statistical method as our short-text algorithm and a k-nearest


neighbor classification method as our long-text algorithm.

Our system has high specificity when dealing with short-text input and high sensitivity when dealing with long-text input. This shows that short-text authentication and long-text authentication are complementary to build a strong authentication system that cannot be easily compromised while remaining convenient and not annoying for the users.

The parameter optimization process that we introduced helped us decide which keystroke features are essential for user identification and which are redundant, and this increases the performance and efficiency and reduces the overhead of the authentication system.

We concluded that typing speed is the biggest parameter that indicates users' typing experience, regardless of their age, gender, or typing errors, and that typing experience influences the performance of our system.

Further investigation is necessary in the following areas: Firstly, different environmental conditions for training and testing must be explored. Secondly, data is required to be collected over long times to decide if users' typing patterns change and whether our system can handle these changes. Thirdly, experiments that evaluate our system over naïve users (users that are unaware that their typing patterns are captured) and imposters that are trying to mimic users' typing pattern are required. Moreover, two different fields should be explored: the first one is implementing and evaluating a keystroke dynamics biometric system for smart phones touch screens keyboards. The other field that requires exploring is implementing and evaluating a keystroke dynamics biometric system for other languages than English.

6. Acknowledgements

We would like to express our gratitude to all those who participated in the experiments.

7. Appendix

Text 1:

This is an Aesop fable about the bat and the weasels.
A bat who fell upon the ground and was caught by a weasel pleaded to be spared his life. The weasel refused, saying that he was by nature the enemy of all birds. The bat assured him he was not a bird, but a mouse, and thus was set free.
Shortly afterwards, the again fell to the ground and was caught by another weasel, whom he likewise entreated not to eat him. The weasel said that he had a special hostility to mice. The bat assured him that he was not a mouse, but a bat, and thus a second time escaped.
The moral of the story: it is wise to turn circumstances to good account.

Text 2:

This is an Aesop fable about the wolf and the crane.
A wolf who had a bone stuck in his throat hired a crane, for a large sum, to put his head into his mouth and draw out the bone. When the crane extracted the bone and demanded the promised payment, the wolf grinding his teeth, exclaimed: "Why, you have surely already had a sufficient recompense, in having been permitted to draw out your head in safety from the mouth and jaws of a wolf"
The moral of the story: in serving the wicked, expect no reward, and be thankful if you escape injury for your pains.

8. References

[1] S. D. Abualgasim and I. Osman. An Application of the Keystroke Dynamics Biometric for Securing PINs and Passwords. World of Computer Science and Information Technology Journal (WCSIT), 1(9):398–404, 2011.

[2] A. A. Ahmed and I. Traore. Biometric Recognition based on Free-text Keystroke Dynamics. IEEE Transactions on Cybernetics, 44(4):458–472, 2014.

[3] G. L. Azevedo, G. D. Cavalcanti, and E. C. Filho. Hybrid Solution for the Feature Selection in Personal Identification Problems through Keystroke Dynamics. International Joint Conference on Neural Networks, pp. 1947–1952, 2007.

[4] K. S. Balagani, V. V. Phoha, A. Ray, and S. Phoha. On the Discriminability of Keystroke Feature Vectors used in Fixed Text Keystroke Authentication. Pattern Recognition Letters, 32(7):1070–1080, 2011.

[5] N. Bartlow and B. Cukic. Evaluating the Reliability of Credential Hardening through Keystroke Dynamics. IEEE 17th International Symposium on Software Reliability Engineering (ISSRE'06), pp. 117–126, 2006.

[6] F. Bergadano, D. Gunetti, and C. Picardi. User Authentication through Keystroke Dynamics. ACM Transactions on Information and System Security (TISSEC), 5(4): 367–397, 2002.

[7] Ceker and S. Upadhyaya. Enhanced Recognition of Keystroke Dynamics using Gaussian Mixture Models, 2015.

[8] G. Dunn and B. S. Everitt, An Introduction to Mathematical Taxonomy. Courier Corporation, 2004.

[9] R. Giot, M. El-Abed, and C. Rosenberger. Greyc Keystroke: A Benchmark for Keystroke Dynamics Biometric Systems. IEEE 3rd International Conference on Biometrics: Theory, Applications, and Systems (BTAS'09), pp. 1–6, 2009.

[10] D. Gunetti, C. Picardi, and G. Ruffo. Dealing with Different Languages and Old Profiles in Keystroke Analysis of Free Text. In AI* IA 2005: Advances in Artificial Intelligence. Springer Berlin Heidelberg, pp. 347–358, 2005.

[11] J. Hu, D. Gingrich, and A. Sentosa. A K-Nearest Neighbor Approach for User Authentication through Biometric Keystroke Dynamics. IEEE International Conference on Communications (ICC'08), pp. 1556–1560, 2008.

[12] D. Jurafsky and J. H. Martin. Speech and Language Processing. Prentice Hall, Englewood Cliffs, New Jersey 7632, 2000.

[13] K. S. Killourhy and R. Maxion. Comparing Anomaly-detection Algorithms for Keystroke Dynamics. In IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'09), pp. 125–134, 2009.

[14] J.-W. Lee, S.-S. Choi, and B.-R. Moon. An Evolutionary Keystroke Authentication based on Ellipsoidal Hypothesis Space. In Proceedings of the 9th ACM Annual Conference on Genetic and Evolutionary Computation, pp. 2090–2097, 2007.

[15] H.-R. Lv, Z.-L. Lin, W.-J. Yin, and J. Dong. Emotion Recognition based on Pressure Sensor Keyboards. IEEE International Conference on Multimedia and Expo, pp. 1089–1092, 2008.

[16] J. Montalvão, C. A. S. Almeida, and E. O. Freire. Equalization of Keystroke Timing histograms for Improved Identification Performance. In IEEE International Telecommunications Symposium, pp. 560–565, 2006.

[17] R. V. Ponkshe and V. Chole. Keystroke and Mouse Dynamics: A Review on Behavioral Biometrics. International Journal of Computer Science and Mobile Computing, 4(2):341–345, 2015.

[18] J. A. Robinson, V. M. Liang, J. A. M. Chambers, and C. L. MacKenzie. Computer User Verification using Login String Keystroke Dynamics. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, 28(2):236–241, 1998.

[19] D. Shanmugapriya and G. Padmavathi. Virtual Key Force – A New Feature for Keystroke. International Journal of Engineering Science and Technology (IJEST), 3(10):7738–7743, 2011.

[20] Y. Sheng, V. V. Phoha, and S. M. Rovnyak. A Parallel Decision Tree-based Method for User Authentication based on Keystroke Patterns. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 35(4):826–833, 2005.


[21] M. Villani, M. Curtin, G. Ngo, J. Simone, H. S. Fort, C. C. Tappert, and S.-H. Cha. Keystroke biometric recognition studies on long-text input over the internet. CSIS, Pace University, 2006.

[22] S. Yong, W. K. Lai, and G. Goghill. Weightless Neural Networks for Typing Biometrics Authentication. In Knowledge-Based Intelligent Information and Engineering Systems, Springer Berlin Heidelberg, pp. 284–293, 2004.

Biographies

Mahmoud AbdelDayem received his B.Sc. degree in Computer and Control Engineering from Faculty of Engineering, Tanta University in 2010. Currently, he is a research scholar and a Master's degree student at Computer and Control Engineering Department, Faculty of Engineering, Tanta University, Egypt, under the supervision of Dr. Hamed Hemeda and Dr. Amany Sarhan. His research interests include Pattern Recognition, Machine Learning, Biometrics and Keystroke Dynamics.

Hamed Hemeda received his B.Sc. and M.Sc. degrees in Computer and Control Engineering from Faculty of Engineering, Tanta University and Faculty of Engineering, Mansoura University in 1995 and 1999, respectively. He was awarded the Ph.D. degree in Computer Engineering from Bretagne-Sud University, France in 2009. He is working now as lecturer at Computer and Control Engineering Department, Tanta University, Egypt. His interests are in the areas of: Human Computer Interaction (HCI) and Computer Networking.

Amany Sarhan received her B.Sc. degree in Electronics Engineering and M.Sc. in Computer Engineering from the Faculty of Engineering, Mansoura University in 1990 and 1997, respectively. She was awarded the Ph.D. degree as a joint research between Tanta University, Egypt and University of Connecticut, USA. She is working now as a full professor and head of Computer and Control Engineering Department, Tanta University, Egypt. Her interests are in the area of: Distributed Systems, Software Restructuring, Object-oriented Databases, and Image and video processing, GPU and Distributed Computations.